The Sparks Foundation - Data Science & Business Analytics Internship

TASK 1 - Prediction using Supervised Machine Learning

- SUBMITTED BY NIRANJAN SAHOO

- GRIPJUNE2022*

Aim of work :In this task it is required to predict the percentage of a student on the basis of number of hours studied using the Linear Regression supervised machine learning algorithm.

STEP 1 - Importing the Libraries and data set

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
url="http://bit.ly/w-data"
data=pd.read_csv(url)
```

In [4]:
```
data.head(5)
```

Out[4]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30

In [5]:
```
data.tail(5)
```

Out[5]:

	Hours	Scores
20	2.7	30
21	4.8	54
22	3.8	35
23	6.9	76
24	7.8	86

In [6]:
```
data.shape
```

Out[6]:
```
(25, 2)
```

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Hours   25 non-null     float64
 1   Scores  25 non-null     int64  
dtypes: float64(1), int64(1)
memory usage: 528.0 bytes
```

In [8]:
```
data.columns
```

Out[8]:
```
Index(['Hours', 'Scores'], dtype='object')
```

In [9]:
```
data.describe().T
```

Out[9]:

	count	mean	std	min	25%	50%	75%	max
Hours	25.0	5.012	2.525094	1.1	2.7	4.8	7.4	9.2
Scores	25.0	51.480	25.286887	17.0	30.0	47.0	75.0	95.0

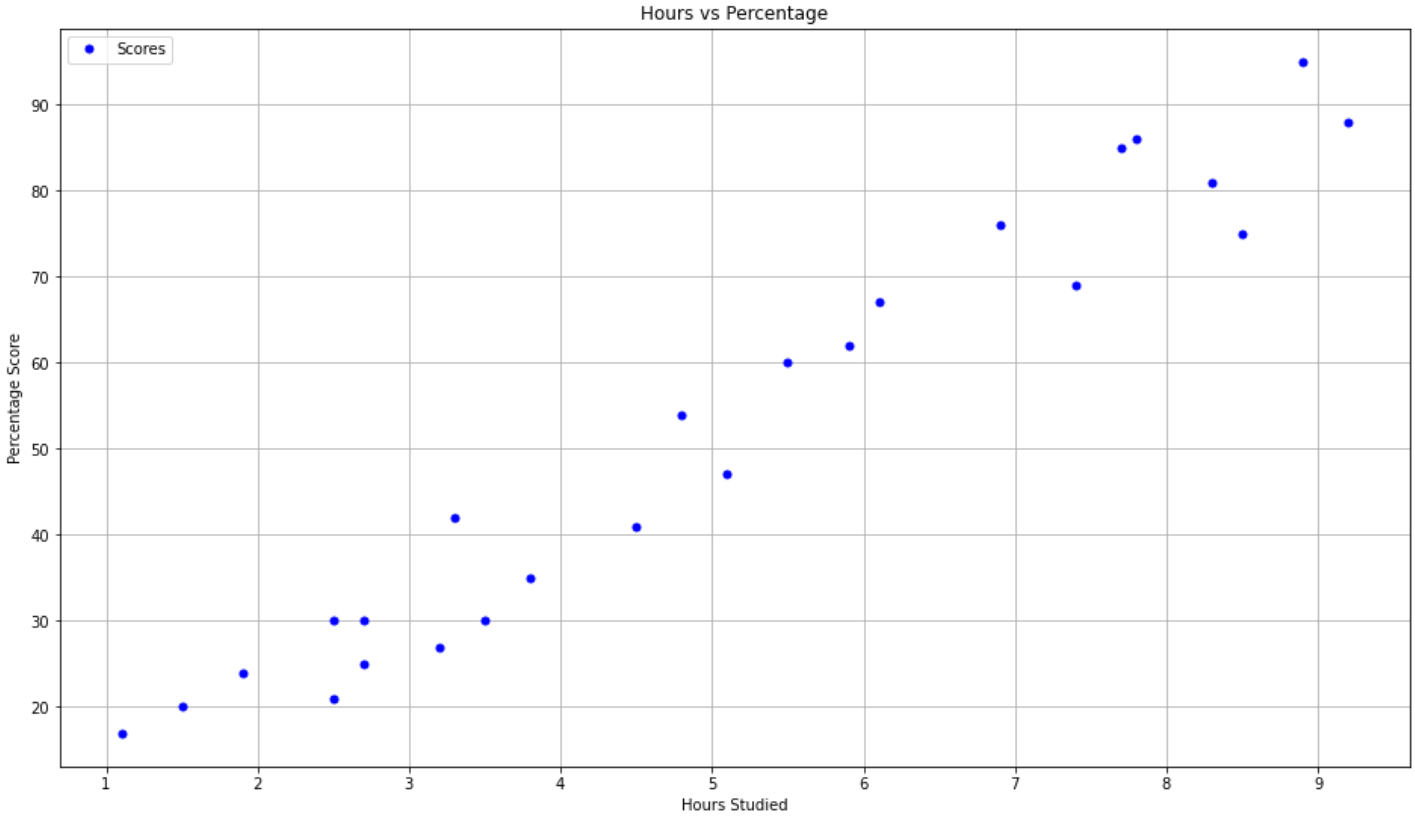
In [10]:
```
# now we will check if our dataset contains null or missings values
data.isnull().sum()
```

Out[10]:
```
Hours     0
Scores    0
dtype: int64
```

There is no nullvalues in our dataset .so we can move to next steps.

STEP 2 - Visualizing the dataset

In [11]:
```
# Plotting the dataset
plt.rcParams["figure.figsize"] = [16,9]
data.plot(x='Hours', y='Scores', style='.', color='blue', markersize=10)
plt.title('Hours vs Percentage')
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.grid()
plt.show()
```



From the graph above, we can observe that there is a linear relationship between "hours studied" and "percentage score". So, we can use the linear regression supervised machine model on it to predict further values.

In [12]:
```
# we can also use .corr to determine the corelation between the variables
data.corr()
```

Out[12]:

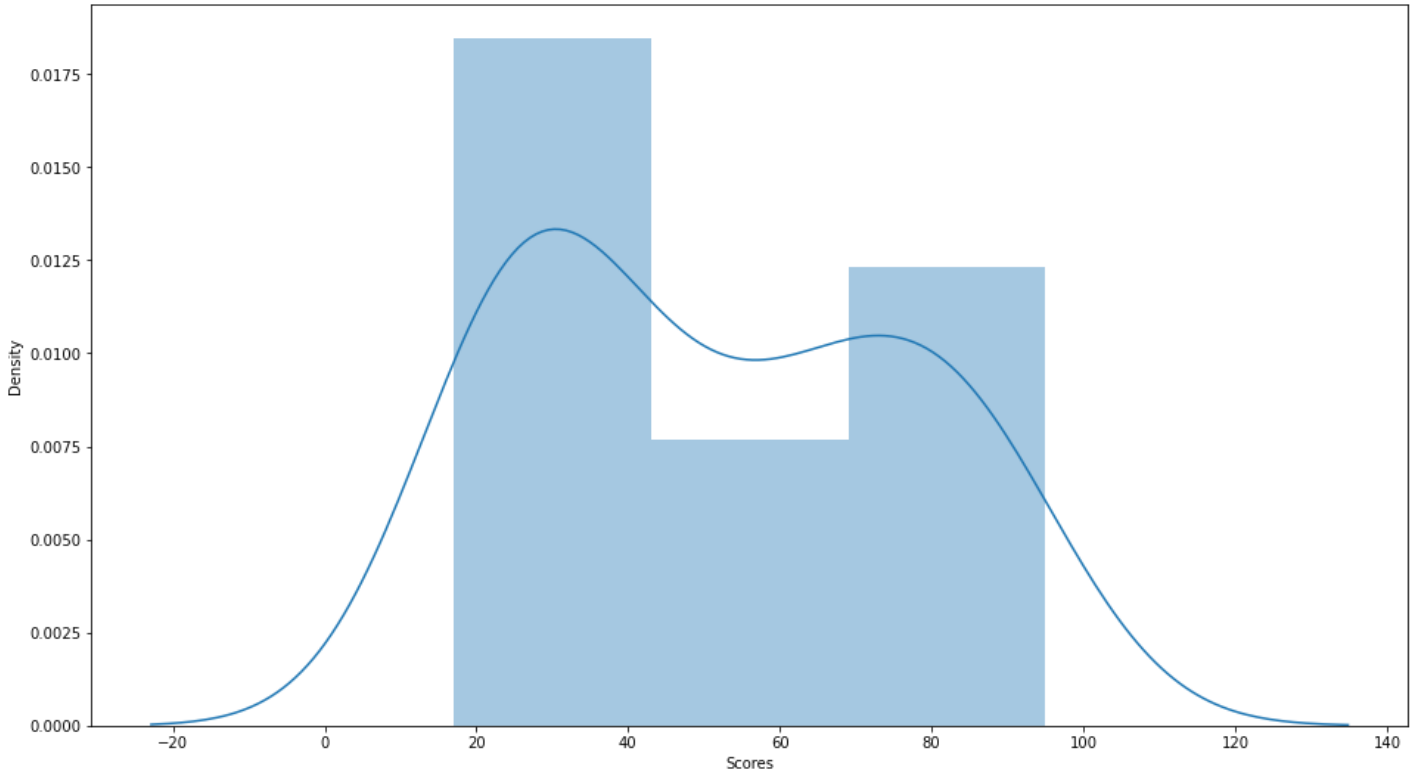
	Hours	Scores
Hours	1.000000	0.976191
Scores	0.976191	1.000000

In [13]:
```
hours=data['Hours']
scores=data['Scores']
```

In [14]:
```
sns.distplot(scores)
```

```
C:\Users\LENOVO\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adapt
your code to use either `displot` (a figure-level function with similar flexibility) or `h
istplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

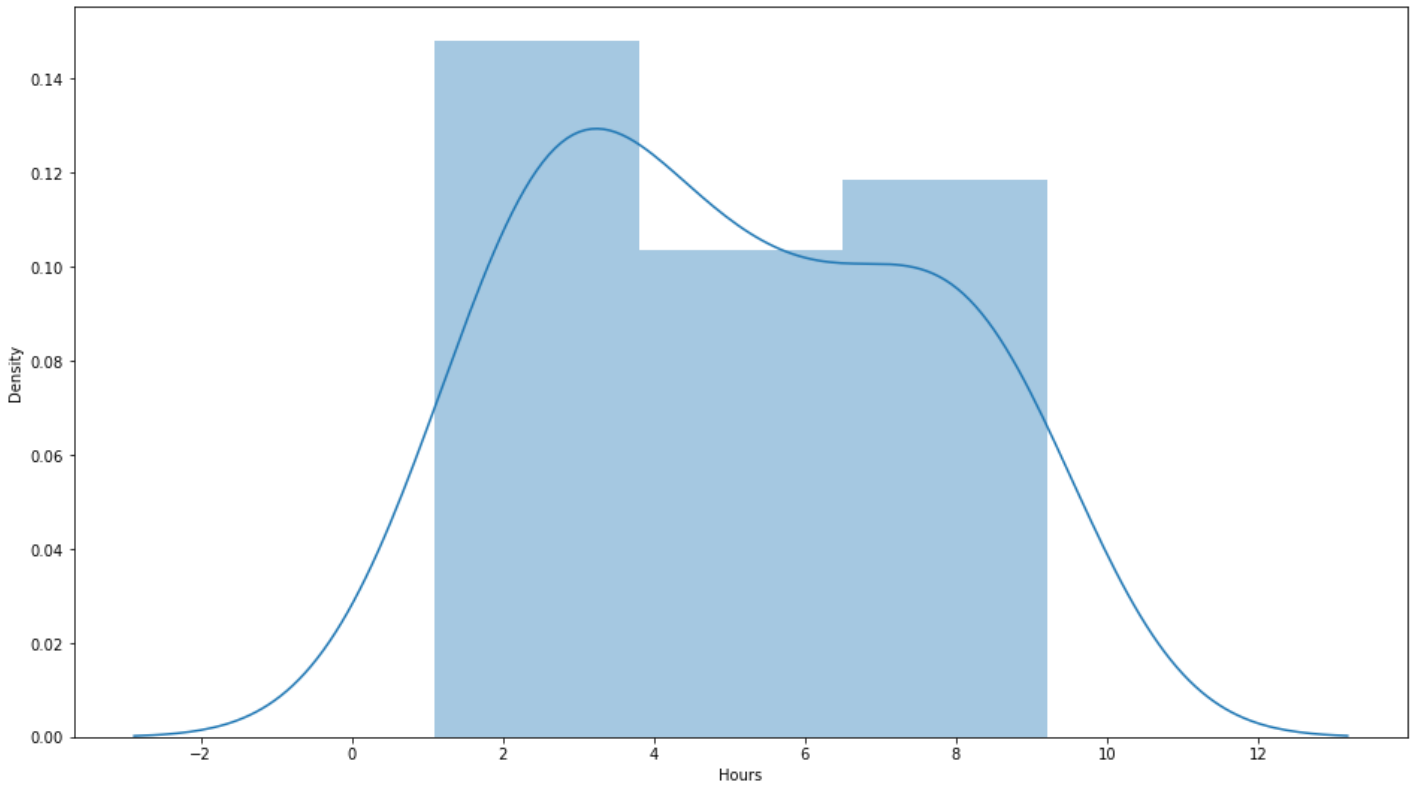
Out[14]:
```
<AxesSubplot:xlabel='Scores', ylabel='Density'>
```



In [15]:
```
sns.distplot(hours)
```

```
C:\Users\LENOVO\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adapt
your code to use either `displot` (a figure-level function with similar flexibility) or `h
istplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[15]:
```
<AxesSubplot:xlabel='Hours', ylabel='Density'>
```



STEP 3 - Data preparation

In this step we will divide the data into "features" (inputs) and "labels" (outputs). After that we will split the whole dataset into 2 parts - testing data and training data.

In [16]:
```
data.head()
```

Out[16]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30

In [17]:
```
# using iloc function we will divide the data
X = data.iloc[:, :1].values
y = data.iloc[:, 1:].values
```

In [18]:
```
X
```

Out[18]:
```
array([[2.5],
       [5.1],
       [3.2],
       [8.5],
       [3.5],
       [1.5],
       [9.2],
       [5.5],
       [8.3],
       [2.7],
       [7.7],
       [5.9],
       [4.5],
       [3.3],
       [1.1],
       [8.9],
       [2.5],
       [1.9],
       [6.1],
       [7.4],
       [2.7],
       [4.8],
       [3.8],
       [6.9],
       [7.8]])
```

In [19]:
```
y
```

Out[19]:
```
array([[21],
       [47],
       [27],
       [75],
       [30],
       [20],
       [88],
       [60],
       [81],
       [25],
```

```
       [85],
       [62],
       [41],
       [42],
       [17],
       [95],
       [30],
       [24],
       [67],
       [69],
       [30],
       [54],
       [35],
       [76],
       [86]], dtype=int64)
```

In [20]:
```
# Splitting data into training and testing data

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

STEP 4 - Training the Algorithm

now we will train our Model.

In [21]:
```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train, y_train)
```

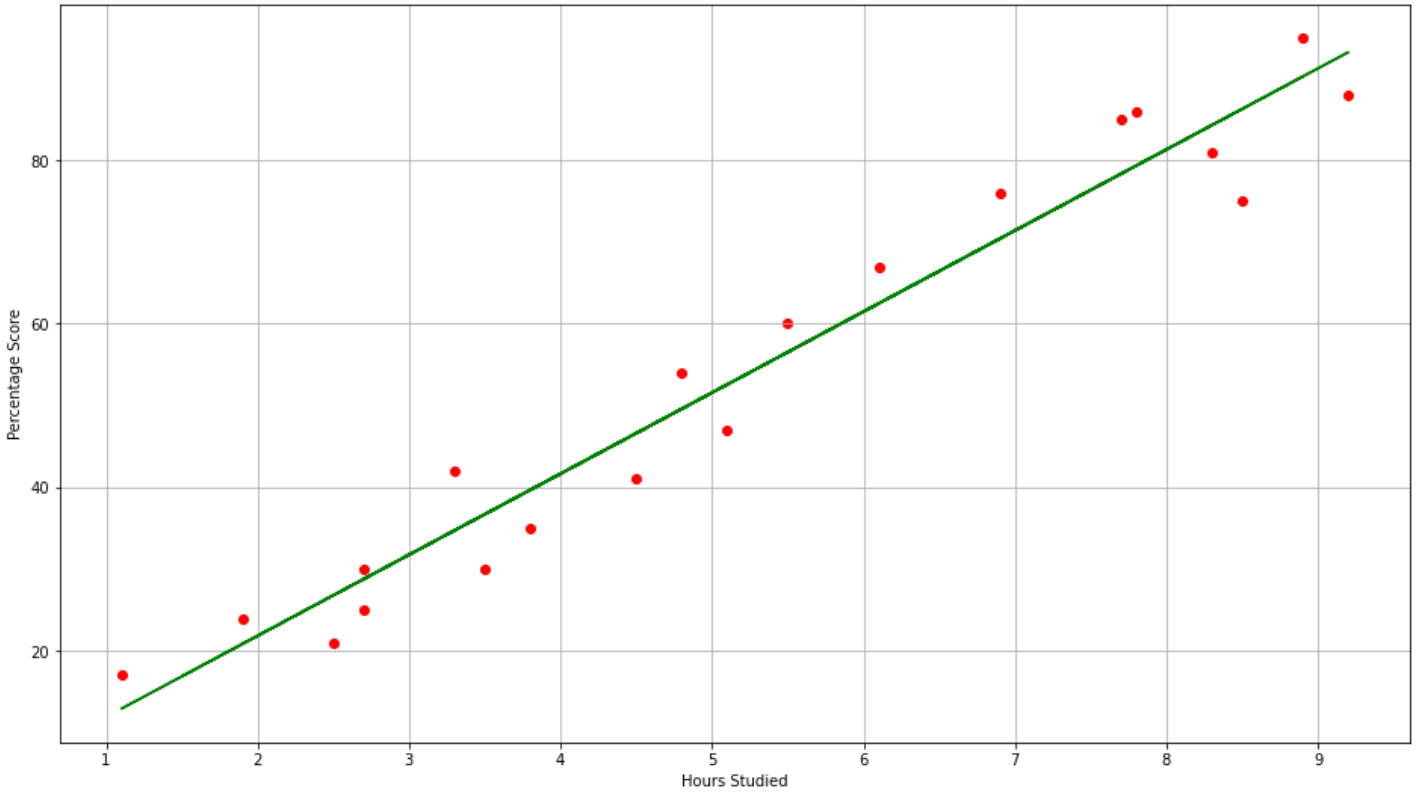
Out[21]:
```
LinearRegression()
```

STEP 5 - Visualizing the model

After training the model, now its time to visualize it.

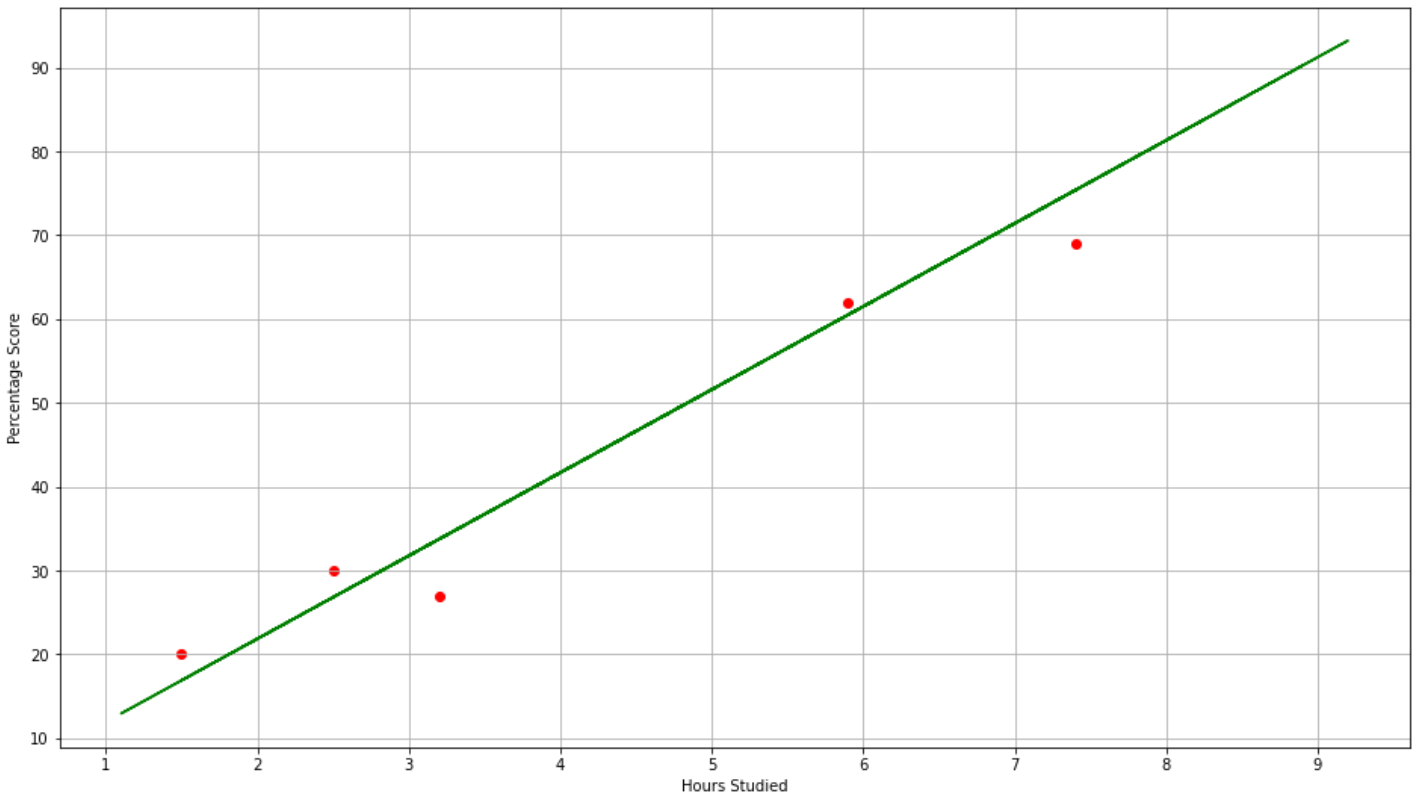
In [22]:
```
line = model.coef_*X + model.intercept_

# Plotting for the training data
plt.rcParams["figure.figsize"] = [16,9]
plt.scatter(X_train, y_train, color='red')
plt.plot(X, line, color='green');
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.grid()
plt.show()
```



In [28]:

```
# Plotting for the testing data
plt.rcParams["figure.figsize"] = [16,9]
plt.scatter(X_test, y_test, color='red')
plt.plot(X, line, color='green');
plt.xlabel('Hours Studied')
plt.ylabel('Percentage Score')
plt.grid()
plt.show()
```



STEP 6 - Making Predictions

Now that we have trained our algorithm, it's time to make some predictions.

In [23]:
```
print(X_test) # Testing data - In Hours
y_pred = model.predict(X_test) # Predicting the scores
```

```
[[1.5]
 [3.2]
 [7.4]
 [2.5]
 [5.9]]
```

In [24]:
```
# Comparing Actual vs Predicted

y_test
```

Out[24]:
```
array([[20],
       [27],
       [69],
       [30],
       [62]], dtype=int64)
```

In [25]:
```
y_pred
```

Out[25]:
```
array([[16.88414476],
       [33.73226078],
       [75.357018  ],
       [26.79480124],
       [60.49103328]])
```

In [26]:
```
# Comparing Actual vs Predicted
comp = pd.DataFrame({ 'Actual':[y_test],'Predicted':[y_pred] })
comp
```

Out[26]:

	Actual	Predicted
0	[[20], [27], [69], [30], [62]]	[[16.884144762398037], [33.73226077948984], [7...

In [27]:
```
# Testing with your own data

hours = 9.25
own_pred = model.predict([[hours]])
print("The predicted score if a person studies for",hours,"hours is",own_pred[0])
```

```
The predicted score if a person studies for 9.25 hours is [93.69173249]
```

STEP 7 - Evaluating the model

Now to evaluate our trained model by calculating mean absolute error

In [28]:
```
from sklearn import metrics

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
```

```
Mean Absolute Error: 4.183859899002975
```

In [ ]:
```

```